A Classification Algorithm Based on Data Clustering and Data Reduction for Intrusion Detection System over Big Data

Qiuhua Wang[1*], Xiaoqin Ouyang[2] and Jiacheng Zhan[2]
[1] School of Cyberspace, Hangzhou Dianzi University, Hangzhou 310018, P. R. China
[e-mail: wangqiuhua@hdu.edu.cn]
[2] School of Comunication Engineering, Hangzhou Dianzi University, Hangzhou 310018, P. R. China
[e-mail: xiaoqin_ouyang@163.com; 663819218@qq.com]
*Corresponding author: Qiuhua Wang

Received September 2, 2018; revised January 29, 2019; accepted February 6, 2019; published July 31, 2019

Abstract

With the rapid development of network, Intrusion Detection System(IDS) plays a more and more important role in network applications. Many data mining algorithms are used to build IDS. However, due to the advent of big data era, massive data are generated. When dealing with large-scale data sets, most data mining algorithms suffer from a high computational burden which makes IDS much less efficient. To build an efficient IDS over big data, we propose a classification algorithm based on data clustering and data reduction. In the training stage, the training data are divided into clusters with similar size by Mini Batch K-Means algorithm, meanwhile, the center of each cluster is used as its index. Then, we select representative instances for each cluster to perform the task of data reduction and use the clusters that consist of representative instances to build a K-Nearest Neighbor(KNN) detection model. In the detection stage, we sort clusters according to the distances between the test sample and cluster indexes, and obtain k nearest clusters where we find k nearest neighbors. Experimental results show that searching neighbors by cluster indexes reduces the computational complexity significantly, and classification with reduced data of representative instances not only improves the efficiency, but also maintains high accuracy.

Keywords: IDS, KNN, Mini Batch K-Means, Representative Instance, Clustering

This work was partially supported by Zhejiang Province Natural Science Foundation (No. LY19F020039), National Natural Science Foundation of China (No.61401128), Zhejiang Province Natural Science Foundation (No. LQ14F020010), Project of Zhejiang Provincial Key Enterprises Institute Construction and Project of Zhejiang Provincial Smart City Regional Synergy Innovation Center (NO. ZXZH1401015).

http://doi.org/10.3837/tiis.2019.07.021 **ISSN : 1976-7277**

1. Introduction

In recent decades, the explosive growth of the Internet has resulted in an increasing number of people using the Internet in their daily life and business. However, it also leads to a large amount of cyber attacks such as Denial of Service (DoS), Probe (Probing) and Spoofing, which means that there is a pressing need for us to take effective measures to defense the cyber attacks and build a safe cyber environment.

IDS is a security measure that is used to detect malicious activities and irregularities in the network and ensure the integrity, confidentiality and availability of information. Its principle is to collect and analyze network traffic data so as to determine whether there is an attack or a violation of the security strategy in the network. Real-time monitoring of the network through IDS can detect intrusion behaviors in time and generate an alarm, thus greatly improving the security of the network system. Hence, how to design an effective and efficient IDS is a great challenge.

Data Mining, which extracts useful knowledge from massive data, is a popular technique for data analysis. Authors in [1] first explored the use of data mining techniques such as association rules and sequence analysis to establish an attack behavior mining model. From then on, embedding data mining technology in the traditional IDS has attracted more and more attention from researchers who focus on intrusion detection [2]-[13]. Data mining mainly includes regression, correlation analysis, classification, clustering and so on, among which clustering and classification receive more attention. Clustering is an important algorithm in data mining. It divides data into multiple clusters according to the similarity between instances. Common clustering algorithms used in IDS mainly include K-Means [4], Self-Organizing Maps(SOM) [5] and Density-Based Spatial Clustering of Applications with Noise(DBSCAN) [6]. Among them, K-Means is the most popular one. The function of classification algorithms in IDS is to build a classification model with a labeled training set, which can be used by IDS to predict whether the test sample is normal or abnormal. KNN [7], [8], decision trees [9], Support Vector Machine(SVM) [10] and naive bayes [11], [12] are common classification algorithms in IDS. Among them, KNN is the most widely used one for its advantages of mature theory, simple implementation and high accuracy. The above mentioned classifiers are all single classifiers that only use one single data mining algorithm. The opposite is hybrid classifiers that combine several data mining algorithms to construct an IDS with better performance. Many hybrid approaches based IDSs have achieved considerable effectiveness [14]-[16]. There are also many improved algorithms that are based on these traditional data mining techniques, and most of them only aim to improve the effectiveness or accuracy [17], [18].

However, with the advent and rapid development of the era of big data, massive amount of network traffic data are generated and these existing IDSs are not capable of dealing with so much data in time. As a crucial component of network security infrastructure, IDS should be both effective and efficient. Effectiveness is evaluated by detection rate, false alarm rate and true positive rate, while the efficiency is measured by the response time during an attack [19], [20]. However, data mining algorithms always suffer from a high computational burden which leads to an increase in response time when confronted with a large amount of training data. To address this shortcoming, some researchers focus on feature reduction and utilize it as a preprocessing stage to reduce the dimension of the feature space, so as to decrease the computational complexity [21]-[26]. For example, a novel SVM with principal component

analysis(PCA), a dimension reduction method, was proposed to reduce the training time in [23]. Authors in [24] proposed a triangle area based KNN approach to reduce features. Authors in [25], [26] proposed a KNN based on cluster centers and nearest neighbors. Nevertheless, feature reduction does not help much when the data set is too large because the time complexity has much more to do with the size of the training data than the feature dimension. Another way to solve the problem is data reduction. Different from feature reduction, data reduction selects some representative instances from the original data and uses them to replace the original data, which helps reduce the computational complexity significantly like [20]. Besides, an IDS that used cluster-based KNN classifier was proposed in [19], which can be seen as a hybrid classifier. Its main idea is to divide the original data into several clusters with a K-Means-based clustering algorithm and to obtain the centers of the clusters. This scheme predicts the test sample with KNN classifier based on these centers rather than the original data. However, K-Means also faces the problem of time consumption when dealing with big data.

To solve the above mentioned problems, we propose a classification algorithm based on data clustering and data reduction. Our main goal is to build an IDS over big data. In our proposed scheme, data clustering is based on Mini Batch K-Means [27], [28], and data reduction is performed by representative instance selection. Firstly, we use Mini Batch K-Means to divide the normal dataset and attack dataset into clusters with similar size separately and the center of each cluster is used as the cluster index. Then, we propose a novel method to select representative instances from each cluster. The representativeness of an instance is related to both density and distance. Higher representativeness makes an instance more likely to be chosen as a representative instance. After selection, we assign a weight to each representative instance. This step not only reduces the size of the original data, but also preserves its maximum amount of information. At the stage of detection, KNN is employed to classify the test sample. We first calculate the distances between the test sample and the cluster indexes to obtain k nearest clusters. Then, we search k nearest neighbors from the representative instances of k nearest clusters to vote on the class of the test sample. Finally, we examine the effectiveness and efficiency of our proposed IDS with the full dataset of Knowledge Discovery and Data Mining CUP99(KDDCUP99) [29].

Our contributions in this paper are as follows.

(1) We use Mini Batch K-Means for data clustering which is suitable for big data. The efficiency is increased obviously compared with the common used K-Means.
(2) We propose a novel method based on distance and density to select representative instances from each cluster, which not only reduces the size of the original data significantly but also keeps the original information to a maximum.
(3) We obtain the representative instances of k nearest clusters by computing the distances between the test sample and cluster indexes, and find k nearest neighbors from them. The way we search k neighbors is proved to be both effective and efficient through the experiments.
(4) The total time taken by our proposed scheme is greatly reduced compared with the traditional KNN under the same conditions, which means the efficiency of our proposed scheme is greatly improved.

The rest of this paper is organized as follows. Section 2 introduces some basic relevant algorithms related to the proposed idea. In Section 3, we give a detailed description of the design and implementation of the proposed classification algorithm. Section 4 discusses the experimental results of our proposed IDS. Finally, we conclude the paper in Section 5.

2. Theoretical Basis

In this section, we mainly explain the problem formulation and introduce the relevant algorithms: Mini Batch K-Means and KNN.

2.1 Problem Formulation

We aim to solve the problem of low efficiency of IDS when faced with big data. We view IDS as a classification problem that divides the input data extracted from the network flow into two classes, thus normality and attack. Let $D=\{(x_1,y_1),(x_2,y_2),...,(x_n,y_n)\}$ be the labeled training data, where n is the size of D, and $x_i \in R^d (1 \le i \le n)$ is an instance in D which can be defined over d-dimension feature space, such as $x_i=\{x_i^{\ 1},x_i^{\ 2},...,x_i^{\ d}\}$, d is the number of features in x_i. Let $y_i \in \{-1,+1\}$ be the label of each instance. $y_i=-1$ means that x_i is normal, and $y_i=+1$ means that x_i is abnormal. The major work in this paper is to train a classifier with D, and use it to determine whether the test sample is normal.

2.2 Mini Batch K-Means

Mini Batch K-Means is a clustering algorithm that yields excellent clustering results with low computation cost on large data sets [28]. For a given dataset D, the goal of this algorithm is to find the set C to minimize the following function:

$$\min \sum_{x \in D} \left\| f(C,x)-x \right\|^2 \tag{1}$$

, where $f(C,x)$ returns the nearest cluster center $c \in R^d$ to x, $|C|=K$, and K is the number of clusters we expect to find. The distance between any two instances x_a and x_b is calculated according to (2) [30].

$$distance(x_a,x_b)=\sqrt{\sum_{j=1}^{d}(x_a^{\ j}-x_b^{\ j})^2} \tag{2}$$

The detail of Mini Batch K-Means is given in **Algorithm 1**.

Different from K-Means, for each iteration, Mini Batch K-Means only selects a subset of instances randomly from the original data instead of using the whole data set(Line 4), which greatly reduces the convergence time. Besides, we use per-center learning rates for fast convergence, in the manner of [31] (Line 9 to Line 12).

2.3 K-Nearest Neighbor

KNN is one of the most simple and common used techniques to classify samples [32]. An instance is classified by a majority vote of its neighbors. In other words, each instance is assigned to the class most common among its k nearest neighbors, as illustrated in **Algorithm 2**.

Obviously, the greater the size of D is, the more time the KNN takes. Although the algorithm is attractive due to its mature theory and good performance, it still makes many researchers disturbed because of its high computational complexity when faced with big data.

Algorithm 1 Mini Batch K-Means

Require: K, batch size b, iteration t, given data D
Ensure: centers C

```
Initialize each c ∈ C with x randomly picked from D
v ← 0
for i = 0 to t do
    M ← b instances picked randomly from D
    for x ∈ M do
        d[x] ← f(C, x) //Cached the center nearest to x
    end for
    for x ∈ M do
        c ← d[x] //Get cached center for this x
        v[c] ← v[c] + 1 //Update each center counts
        η ← 1/v[c] //Get per-center learning rate
        c ← (1 − η)c + ηx //Take gradient step
    end for
end for
return C
```

Algorithm 2 KNN Algorithm

Require: k, given data D, test sample s
Ensure: label of s

```
dist ← []
for i = 0 to D.count do
    dist[i] ← distance(D[i], s)
end for
k_neighbors ← obtain k nearest instances from D
count the occurrences of labels among k_neighbors
return label with maximum occurrence
```

3. Our Proposed Scheme

In this section, our proposed method is presented. It mainly consists of three steps: data clustering, data reduction and KNN classification. We take data clustering and data reduction as the training stage and KNN classification as the detection stage.

3.1 Data clustering based on Mini Batch K-Means

We first classify training data into two classes according to their labels. We use *ND* to denote normal instances, and *AD* to denote attack instances. Then, Mini Batch K-Means is used to divide *ND* and *AD* into several clusters with similar size separately. **Algorithm 3** describes the detail about data clustering. The number of original clusters we expect to find is calculated according to (3)(Line 1). *Clus* is the clusters obtained from Mini Batch K-Means, and *Cens* is the centers of *Clus* (Line 2). For each cluster in *Clus* , if its size is smaller than *minV* (the minimum size of data permitted in one cluster), the instances in this cluster are assigned to the

nearest cluster (Line 5 to Line 6); If its size is larger than $maxV$ (the maximum size of data permitted in one cluster), **Algorithm 3** is called again to divide this cluster into smaller clusters with permitted size(Line 8 to Line 10). Finally, the sizes of clusters we obtained are all between $minV$ and $maxV$.

$$oriCluster = \frac{size}{maxV} + 1 \tag{3}$$

, where *size* is the number of instances.

Algorithm 3 ClusterData

Require: $minV$, $maxV$, given data D
Ensure: $Clus, Cens$
1: $oriClus \leftarrow$ calculate the number of clusters we want according to (3)
2: $Clus, Cens \leftarrow MiniBatchKMeans()$
3: **for** $i = 0$ to $oriClusters$ **do**
4: **if** $Clus[i].count \leq minV$ **then**
5: $closestClus \leftarrow findNearestCluster()$
6: add items in this cluster to $closestCluster$
7: **else**
8: $newClus, newCens \leftarrow ClusterData(minV, maxV, Clus[i])$
9: add $newCens$ to $Cens$
10: add $newClus$ to $Clus$
11: **end if**
12: delete $Cens[i]$
13: delete $Clus[i]$
14: **end for**
15: **return** $Clus, Cens$

3.2 Data Reduction

The purpose of this step is to reduce the original data by selecting representative instances from each cluster. The way we select representative instances refers to the initial centers selection method of K-Means++ algorithm [33]. The procedure of this step can be seen in **Algorithm 4**. The main factors of an instance to be selected as representative are distance and density. We use *X* to denote all data in one cluster and *S* to save the selected representative instances. For each instance, we take the number of its neighbors whose distances to it are less than ε as its density. ε can be obtained by (4).

$$\varepsilon = \frac{1}{2} \times \max_{x \in X} distance(x, c) \tag{4}$$

, where *c* denotes the center of *X*.

As a result, we need to calculate the ε of *X* and the density of each instance before selection(Line 2 to Line 4). First of all, we calculate the number of instances *m* we need to select from each cluster with *n* instances according to (5)(Line 5).

$$m = \alpha \times (1 - e^{-\frac{n}{1000}}) \tag{5}$$

, where α is a coefficient for instances selection.

Then, we choose an instance at random from *X* and save it to *S* as the first representative

Algorithm 4 Representative instances selection

Require: size of cluster n, coefficient α, given data X
Ensure: representative instances S, weight of each selected instances W

1: $\epsilon \leftarrow$ calculate according to 4
2: **for** $x \in X$ **do**
3: $density[x] \leftarrow$ the number of the neighbors whose distances to x are less than ϵ
4: **end for**
5: $m \leftarrow$ calculate number of representative instances according to 5
6: $S[0] \leftarrow x$ picked from X randomly
7: **for** $j = 2$ to m **do**
8: **for** $i = 1$ to n **do**
9: $d[i] \leftarrow findNearestDistance(X[i], S)$
10: **end for**
11: **for** $x \in X$ **do**
12: $P[x] \leftarrow$ calculate probability of each instance according to 7
13: **end for**
14: $S[j] \leftarrow$select an instance based on probability P
15: **end for**
16: **for** $j = 1$ to m **do**
17: $radius \leftarrow findNearestDistance(S[j], S)$
18: $W[j] \leftarrow 0$
19: **for** $i = 0$ to n **do**
20: **if** $distance(S[j], X[i]) < radius$ **then**
21: $W[j] \leftarrow W[j] + 1$
22: **end if**
23: **end for**
24: **end for**
25: **return** S, W

instance(Line 6). After that, we continue to select the rest *m*-1 representative instances. During each selection, for each instance $x_i \in X$, we obtain the nearest distance between *x* and the instances that have already been chosen just like (6) (Line 8 to Line 10).

$$d_{\min(x,S)} = \min_{\forall s \in S} distance(x, s) \tag{6}$$

The rest representative instances are selected based on a weighted probability distribution. An instance *x* is chosen with a probability that is related to both $d_{\min(x,S)}$ and $density[x]$, which is calculated according to (7)(Line 11 to Line 13).

$$P(x) = \frac{\dfrac{d^2_{\min(x,S)}}{\sum_{x \in X} d^2_{\min(x,S)}} \times e^{-\frac{density[x]}{n}}}{Z} \tag{7}$$

Z is the normalization factor, which is obtained by (8). It makes sure that $1 \le P(x) \le 1$ and $\sum_{x \in X} P(x) = 1$.

$$Z = \sum_{x \in X} \frac{d^2_{\min(x,S)}}{\sum_{x \in X} d^2_{\min(x,S)}} \times e^{-\frac{density[x]}{n}} \tag{8}$$

After *m* representative instances are selected, we assign a weight to each selected instance in *S*. The weight of *s* is determined by the minimum distance between itself and the other instances in *S*, as shown in (9)(Line 17). Afterwards, we traverse every instance in *S* and count the instances in *X* whose distances to *s* are less than $d_{\min(x,S)}$, the result of which is considered as the weight of *s*(Line 18 to Line 23).

$$d_{\min(s,S)} = \min_{s_1 \in S, s_1 \neq s} distance(s, s_1) \tag{9}$$

The main idea of this method is to make the representative instances distribute uniformly. The aim of assigning a weight to each selected instance is to help retain the original information, even the size of original data is reduced greatly.

3.3 KNN Classification

This step is to classify the test sample into normality or attack. **Algorithm 5** shows the detail of this procedure. *inst* denotes a test sample. *k* denotes the number of neighbors we need. *CS* denotes the clusters that consist of representative instances and CI denotes the indexes of *CS*. *CW* denotes the weights corresponding to *CS*. Each CW_i consists of the weights of instances in CS_i.

We first sort *CS* according to the distances from *inst* to CI (Line 1 to Line 4). Then, we find *k* nearest clusters and search *k* nearest neighbors from them (Line 5 to Line 6). Finally, we use the weighted KNN algorithm for classification. We add up the weights of neighbors with the same label and take the label with the maximum sum of weights as the label for *inst*. Since only binary classification is considered here, we just need to add up the weights of all neighbors and return the positive and negative of the sum of weights (Line 7 to Line 12).

Algorithm 5 KNN Classification

Require: k, CI, CS, W, $inst$
Ensure: label of $inst$
1: **for** $i = 0$ to $CI.count$ **do**
2: $d[i] \leftarrow calDistance(CI[i], inst)$
3: **end for**
4: $CS' \leftarrow$sort CS based on d
5: $nearClus \leftarrow$ get k nearest clusters from CS'
6: $k\_neighbors \leftarrow$ find k nearest neighbors from $nearClus$
7: $weight \leftarrow$ add up the weights of $k\_neighbors$
8: **if** $weight \geq 0$ **then**
9: **return** +1
10: **else**
11: **return** -1
12: **end if**

3.4 Process of the Proposed Classification Algorithm

Since we have introduced all major steps of our proposed algorithm in detail, we conclude the process of our proposed classification algorithm in this section.

The original data is shown in **Fig. 1**. " ● " denotes an attack instance whose label is +1, and " ○ " denotes a normal instance whose label is -1. **Fig. 2** illustrates the result after the step of data clustering which is performed by **Algorithm 3**. " ▼ " denotes the center of the cluster that consists of attack instances, and " ✶ " denotes the center of the cluster that consists of normal instances. The red circles denote the clusters obtained by data clustering. **Fig. 3** shows the result after the step of data reduction that is performed by **Algorithm 4**. Data in each cluster is reduced by representative instance selection algorithm.

We assume that " □ " denotes a test sample and we employ **Algorithm 5** to predict whether the sample is normal. Here we set the number of neighbors k=4. As shown in **Fig. 4**, 4 nearest centers are found by calculating the distances from the test sample to cluster centers. **Fig. 5** shows the representative instances of the 4 clusters we found before. Then we search 4 nearest neighbors from these instances and use them to vote on the class of the test sample.

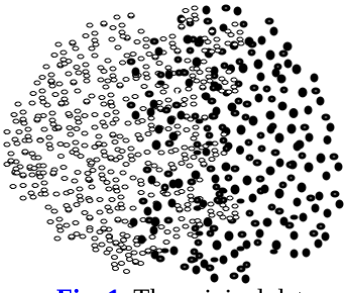


Fig. 1. The original data

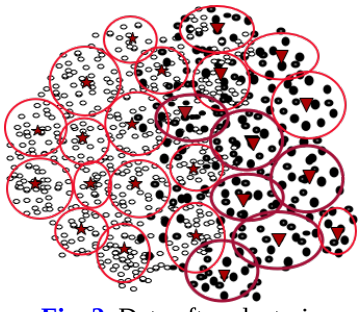


Fig. 2. Data after clustering

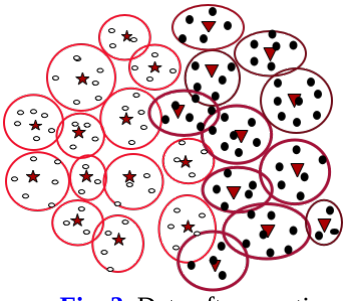


Fig. 3. Data after reuction

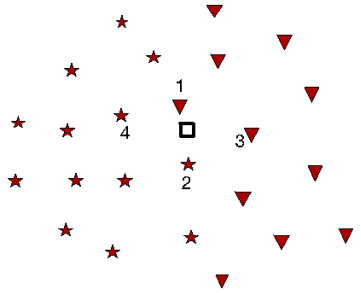


Fig. 4. Cluster indexes

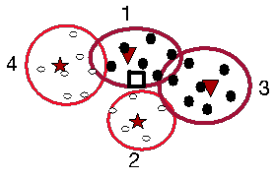


Fig. 5. Nearest clusters

4. Experimental Results and Performance Analysis

4.1 Experimental Environment

To prove the performance of our proposed scheme, we conducted a series of experiments with the full dataset of KDDCUP99. Since 1999, KDDCUP99 has been the most widely used data set for evaluating the anomaly detection methods, which is built based on the data captured in Defense Advanced Research Projects Agency1998(DARPA1998) IDS evaluation program. KDDCUP99 consists of approximately 4,900,000 single connection vectors. Most researchers used 10% of KDDCUP99 to perform experiments, while we used the full data of it to evaluate the performance of our scheme when confronting with large scale of data. Each instance of KDDCUP99 contains 41 features and is labeled as normal or abnormal [29]. The abnormal instances are divided into four classes: probing, denial of service (DoS), remote to local (R2L) and user to root (U2R). As too many duplicate samples exist in KDDCUP99, before

performing the experiments, we removed the duplicate samples in the dataset. The details of KDDCUP99 after de-duplication are shown in **Table 1**.

Table 1. Details of KDDCUP99 after de-duplication

Class	Normal	Probing	DoS	R2L	U2R	All
Number of instances	812814	13860	247267	999	43	1074983
Percentage(%)	75.61	1.29	23	0.09	0.004	100

We use 10-fold cross validation to divide the dataset into ten subsets and perform ten experiments. For each experiment, one subset is used as test sample set and the rest nine subsets are used as training data sets. The mean result of ten experiments is taken as the final result.

The operating system we use in simulation is Windows 10 with 2.2GHz CPU and 8G memory, and the programming language used to implement these experiments is Python 3.6.

4.2 Evaluation Indicator

(1) Clustering Indicator

In this section, the evaluation indicators about clustering are given. The clustering time is used to evaluate the efficiency. The shorter the better. And the Sum of the Squared Error (*SSE*) is used to evaluate the quality. The smaller the better.

For a given clustered data *X*, its centers are denoted by $C = \{C_1, C_2, ..., C_K\}$, where *K* is the number of clusters. SSE is obtained by (10). $x \in C_i$ means that *x* belongs to the i_{th} cluster.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} distance(x, C_i)^2 \tag{10}$$

(2) Classification Performance Indicator

As mentioned earlier, effectiveness and efficiency are both important to an IDS. For efficiency, classification time is an obvious indicator. As to effectiveness, Accuracy(ACC), True Positive Rate(TPR) and False Positive Rate(FPR) are used as its indicators. Before we explain ACC, TPR and FPR, we introduce True Positive(TP), True Negative(TN), False Positive(FP) and False Negative(FN) first.

TP denotes the number of abnormal instances that are identified as abnormal. TN denotes the number of normal instances that are identified as normal. FP denotes the number of normal instances that are identified as abnormal. FN denotes the number of abnormal instances that are identified as normal.

As defined in [34] and [35], $ACC = \frac{TN + TP}{TN + FN + TP + FP}$ represents the proportion of correctly predicted instances to total. $TPR = \frac{TP}{FN + TP}$ represents the proportion of instances which are correctly predicted as attack to all attack instances. $FPR = \frac{FP}{TN + FP}$ represents the proportion of instances which are incorrectly predicted as attack to all normal instances.

4.3 Clustering performance with Mini Batch K-Means

We use Mini Batch K-Means instead of K-Means to cluster data. **Fig. 6** and **Fig. 7** show the reasons. **Fig. 6** gives the clustering time of Mini Batch K-Means and K-Means, and **Fig. 7**

shows the SSE of them.

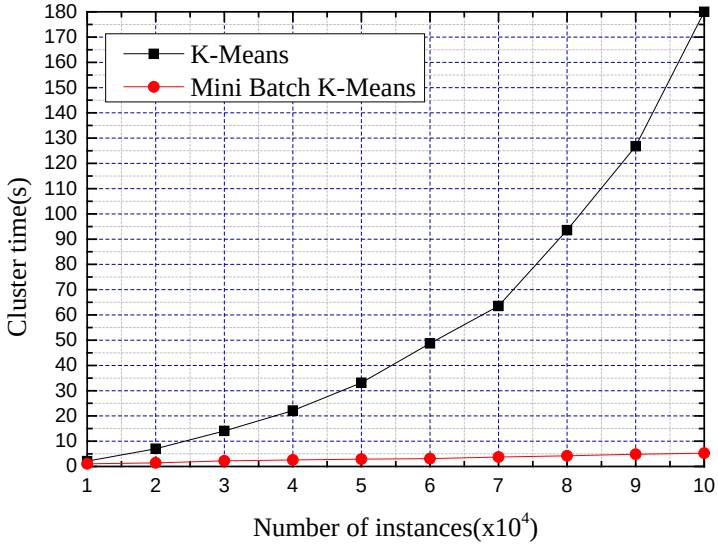


Fig. 6. Time for clustering

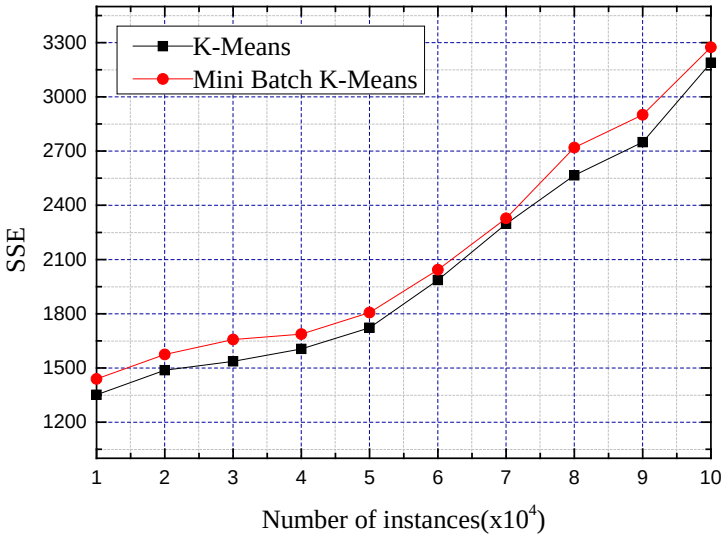


Fig. 7. SSE of clustering

From **Fig. 6**, we can see that the clustering time taken by K-Means is much longer than Mini Batch K-Means when dealing with the same data. Besides, the difference becomes greater with the increase of the size of data. On the other hand, **Fig. 7** indicates that the difference of SSE is not so obvious. The clustering quality of Mini Batch K-Means is only slightly worse than K-Means. Taking both of these two indicators into consideration, Mini Batch K-Means is a better choice.

4.4 Classification Performance

The effectiveness of experiments with different value of k is shown in **Table 2**. k is the number of neighbors we use in the stage of detection. As we can see, smaller k leads to higher ACC and TPR. Thus, we set k=1 in later experiments.

Table 2. Effectiveness with different values of *k*

k	ACC(%)	TPR(%)	FPR(%)
1	99.74	99.51	0.53
3	99.71	99.49	0.53
5	99.67	99.24	0.49
10	99.64	99.09	0.48
20	99.50	98.32	0.29

Fig. 8 illustrates the time for data reduction and **Fig. 9** illustrates ACC with increasing $maxV$ and α . As is shown in **Fig. 8**, the greater the $maxV$ or α is, the longer it takes for data reduction. It also can be inferred from (3) and (5) that greater $maxV$ and α lead to more representative instances to be selected and more instances to select from, both of which result in higher computational complexity.

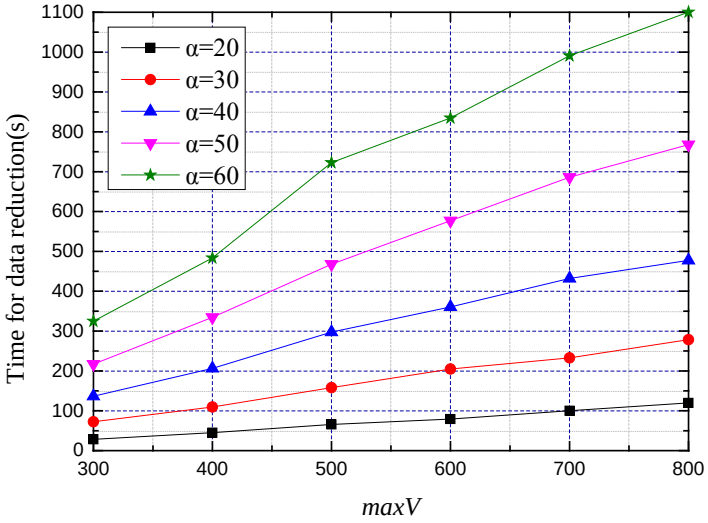


Fig. 8. Time for data reduction with different $maxV$ and α

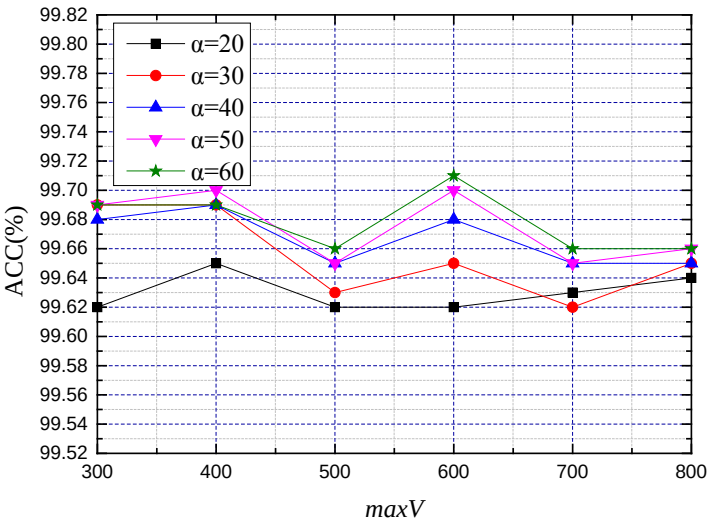


Fig. 9. ACC with different $maxV$ and α

Fig. 9 tells us that $maxV$ has little to do with ACC while α affects to a certain extent, so we perform the experiments with specified $maxV = 600$ and different α from 40 to 90 in order to figure out the specific effect of different α on aspects of training time, detection time and ACC. The results are shown in **Table 3**.

Table 3. Time and ACC with different α when $maxV = 600$

α	ACC(%)	Training time(s)	Detection time(s)
40	99.646	1026.98	74.39
50	99.656	1493.62	95.49
60	99.664	1998.99	104.43
70	99.669	2624.52	115.12
80	99.67	3380.38	128.15
90	99.672	4039.03	140.55

As shown in **Table 3**, greater α brings a little better performance on ACC, but causes much more time for training and detection. However, it also can be indicated that the ACC grows more and more slowly, while the training time grows faster. So we set $\alpha = 70$ where ACC is considerable.

To prove our improvement in efficiency compared with KNN, we performed experiments with KNN and our proposed scheme to make a comparison. **Fig. 10** illustrates the comparison of total time taken by KNN and our proposed scheme.

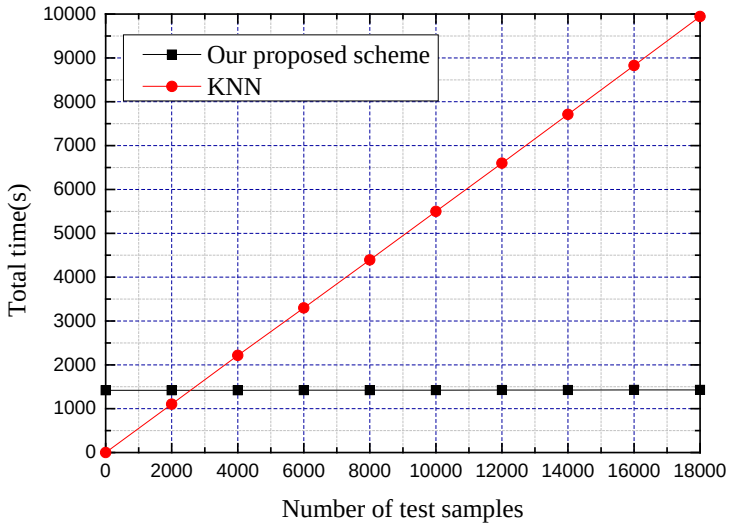


Fig. 10. Comparison of total time between KNN and our proposed scheme

As we can see in **Fig. 10**, when the number of test samples is 0, KNN takes no time while our scheme takes minutes for training. But we can also see that the total time taken by KNN grows quickly while the total time taken by our proposed scheme grows much more slowly. Moreover, when the size of test sample set exceeds a certain threshold, the total time taken by our scheme is much less than KNN, and the disparity between them becomes greater with the increase of the number of test samples. For example, when the number of test samples is 8000, KNN takes 4394.08s while our scheme only takes 1422.64s. When the number of test samples is 16000, KNN takes 8827.16s while our scheme only takes 1427.39s. Thus, our proposed scheme is more efficient than KNN since it takes much less total time under the same condition.

4.5 Performance Comparison and Analysis

(1) Baseline Classifiers

Considering that our scheme is based on KNN, so KNN is considered for comparisons. In addition, some related literatures [24]-[26] also aim to reduce the time complexity and make improvements on KNN which are similar to our proposed scheme. TANN in [24], CANN in [25] and DCNN in [26] are all based on K-Means and KNN algorithms, they reduce the computational complexity by transforming high dimensional data into low dimensional data. They are also considered for make comparisons. In all, there are five schemes are performed to make comparisons.

- Scheme 1 is performed with KNN.
- Scheme 2 is performed with TANN [24].
- Scheme 3 is performed with CANN in [25].
- Scheme 4 is performed with DCNN in [26].
- Scheme 5 is performed with our proposed scheme.

(2) Comparisons of Results

Table 4 shows the detection results of the above five schemes in five types of attack. As we can see, all schemes have good performance for normal, probing and DoS types, but perform poorly in the sparse U2R and R2L types. Among these five schemes, KNN performs best in each type of attack. The result of our proposed scheme is better than the other three schemes, and closest to KNN. Compared with TANN, CANN and DCNN, our proposed scheme shows better performance. Compared with KNN, though our proposed scheme does no better in the view of all types, but the difference between them is not obvious. As a result, the detection result of our proposed scheme is considerable.

Table 4. Comparing detection results of each class for five schemes

Scheme	Normal(%)	Probing(%)	DoS(%)	U2R(%)	R2L(%)
KNN	99.68	98.49	99.98	67.31	91.74
TANN	97.01	94.89	90.94	60	80.53
CANN	97.04	87.61	99.68	63.85	57.02
DCNN	98.69	97.44	93.2	67.31	86.55
Our proposed scheme	99.72	95.54	99.88	64.4	85

Table 5 describes the ACC, TPR, and FPR of the five schemes. From **Table 5**, we can see that KNN performs best among these schemes. It has higher accuracy and lower FTP than the other schemes. Although our proposed scheme performs a little worse than KNN, the accuracy of it is just 0.11% lower than KNN. Moreover, compared with TANN CANN and DCNN, our proposed scheme performs better in accuracy.

Table 5. Comparing ACC, TPR and FPR of five schemes

Scheme	ACC(%)	TPR(%)	FPR(%)
KNN	99.83	99.49	2.57
TANN	99.01	99.27	2.99
CANN	99.46	99.28	2.95
DCNN	99.64	99.43	2.69
Our proposed scheme	99.72	99.44	2.53

4.6 Complexity Analysis and Comparison

In this section, we discuss the time complexities of the baseline classifiers we mentioned above and our proposed scheme, then we will make comparisons of them. First of all, we need to define some variables for analysis.

- d is the original dimension of an instance.
- d' is the dimension after feature reduction.
- n is the number of training samples.
- t is the number of iteration used in clustering algorithms.
- b is the batch size used in Mini Batch K-Means.
- K is the number of clusters we obtain in clustering algorithms.
- K_i is the number of instances we select from i_{th} cluster.
- C_i is the i_{th} cluster we get, where $1 \le i \le K$.
- n_i is the number of instances contained in i_{th} cluster, and $n_i \approx n/K$.
- m is the number of test samples.

After variable definition, we analyze the complexities of these schemes. All of TANN, CANN, DCNN and our proposed scheme consist of three steps. Step 1 is about clustering. Step 2 is data reduction or feature reduction. Step 3 is KNN classification. So we discuss the complexities of them from three steps in **Table 6**.

Table 6. Comparing time complexities of five schemes

Scheme	Step 1	Step 2	Step 3
KNN	O(0)	O(0)	O($m*n*d$)
TANN	O($t*n*K*d$)	O($n*K*(K-1)*d$)	O($m*[K*(K\text{-}1)*d+n*d'$)]
CANN	O($t*n*K*d$)	O($n*(K+n_i)*d$)	O($m*[(K+n_i)*d+n*d'$)]
DCNN	O($t*n*K*d$)	O($n*(K+n_i)*d$)	O($m*[(K+n_i)*d+n*d'$)]
Our proposed scheme	O($t*b*K*d$)	O($K*K_i*n_i*d$)	O($m*(K+K_i)*d$)

As we can see in **Table 6**, the complexity of KNN grows linearly with the increase of m, n and d. The following four schemes aim to improve KNN's shortcoming of high computational complexity when confronting with big data.

TANN, CANN and DCNN achieve the goal by feature reduction. They transform the d dimensional vector into d' dimensional vector. TANN makes d' =10, CANN makes d' =1, and DCNN makes d' =2. After feature reduction, they perform the same with KNN. The difference between them is that KNN handles higher dimensional data than TANN, CANN and DCNN. As a result, TANN, CANN and DCNN need to take some time to reduce features while KNN needs no time for it. But when m or n exceeds a certain threshold, they can achieve higher efficiency than KNN when dealing with the same dataset.

However, our proposed scheme is different from TANN, CANN and DCNN. We use clustering algorithm and reduce the number of instances in each cluster to find nearest neighbors much faster, instead of reducing the dimension of each instance.

Now we use variable control method to analyze the efficiency of the mentioned schemes when dealing with the same training data and test data. Firstly, we assign values to variables except m. We set $t=100$, $n=500000$ and $d=41$ which is the dimension of instances in

KDDCUP99. TANN, CANN and DCNN set K=5, so $n_i \approx n/K = n/5 = 100000$. Our proposed scheme set K=1000, $b = n/200$, $n_i \approx n/1000 = 500$ and $K_i = n_i/10 = n/10^4 = 50$. Then, we substitute these values into **Table 6**, and we can find out the relationship between total time and m. Here we use $f(m)$ to denote the variation of total time. The relationship between $f(m)$ and m is illustrated in **Fig. 11**.

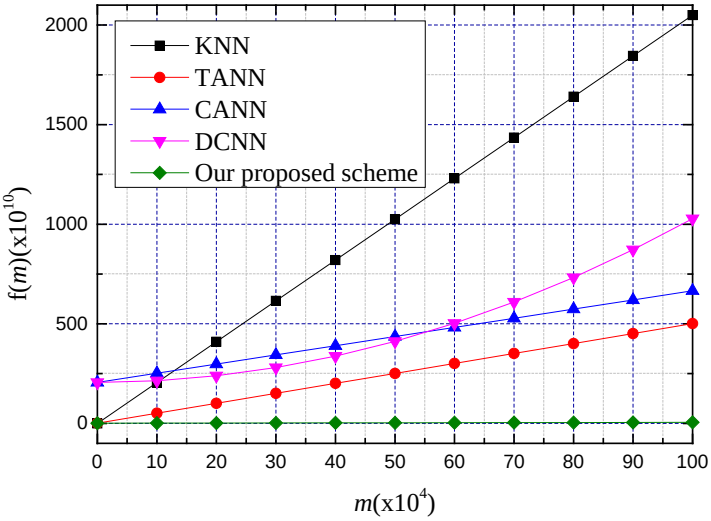


Fig. 11. Total time taken by five schemes with different m

As is shown in **Fig. 11**, when $m = 0$, the time is consumed in the training stage. In the detection stage, time increases with the increase of m. As we can see, KNN has the lowest efficiency. TANN, CANN and DCNN improve a lot, compared with KNN. Among them, TANN takes the least time in the training stage, but its speed in the detection stage is slower than CANN. When m exceeds a certain threshold, the total time taken by TANN will be more than CANN, and the disparity between them becomes greater with the increase of m. As to DCNN, it takes similar time to CANN in the training stage, and its speed in the detection stage is faster than CANN when $m < 550000$. When m exceeds the threshold, the total time taken by DCNN is more than CANN. However, our proposed scheme shows the highest efficiency in both training stage and detection stage.

In conclusion, our proposed scheme performs best. The improvement of it is much more effective compared with TANN, CANN and DCNN. It is suitable for IDS over big data.

5. Conclusion

In this paper, we proposed a classification algorithm based on data clustering and data reduction which is suitable for big data analysis in IDS. We proposed a new method to select representative instances from each cluster for data reduction. In addition, we proposed to search an instance's neighbors by searching its near clusters through computing the distances from the test sample to cluster centers. The above two points both help reduce the computational time significantly while maintain good ACC. We tested our proposed classification algorithm over de-duplicated full data of KDDCUP99 and achieved excellent performance especially in efficiency. We also compared our proposed classification algorithm

with traditional KNN algorithm and three other schemes, the results show that our proposed algorithm performs as well as traditional KNN in effectiveness and much better than the other schemes in efficiency.

In the future, we will further focus on improving the performance of our proposed algorithm, such as optimizing parameters and expanding binary classification to multi-class classification so as to recognize the specific type of attack.

References

[1] W. Lee and S. J. Stolfo, "Data Mining Approaches for Intrusion Detection," in *Proc. of 7th conference on USENIX Security Symposium*, pp. 79-94, Jan. 26-29, 1998. Article (CrossRef Link)

[2] K. Wankhade, S. Patka and R. Thool, "An Overview of Intrusion Detection Based on Data Mining Techniques," in *Proc. of International Conference on Communication Systems and Network Technologies*, pp. 626-629, Apr. 6-8, 2013. Article (CrossRef Link)

[3] E. Ariafar and R. Kiani, "Intrusion Detection System Using An Optimized Framework Based on Data Mining Techniques," in *Proc. of IEEE International Conference on Knowledge-Based Engineering and Innovation*, pp. 785-791, Dec. 22-22, 2017. Article (CrossRef Link)

[4] Z. Li, Y. Li and L. Xu, "Anomaly Intrusion Detection Method Based on K-means Clustering Algorithm With Particle Swarm Optimization," in *Proc. of International Conference of Information Technology, Computer Engineering and Management Sciences*, pp. 157-161, Sept. 24-25, 2011. Article(CrossRef Link)

[5] H. G. Kayacik, A. N. Zincir-Heywood and M. I. Heywood, "On the Capability of an SOM Based Intrusion Detection System," in *Proc. of International Joint Conference on Neural Networks*, pp. 1808-1813, Jul. 20-24, 2003. Article (CrossRef Link)

[6] J. Yang, "An Improved Intrusion Detection Algorithm Based on DBSCAN," *Microcomputer Information*, vol. 25, no. 3, pp. 58-60, Mar. 2009. Article(CrossRef Link)

[7] A. H. Farooqi and A. Munir, "Intrusion Detection System for IP Multimedia Subsystem Using K-nearest Neighbor Classifier," in *Proc. of IEEE International Multitopic Conference*, pp. 423-428, Dec. 23-24, 2008. Article (CrossRef Link)

[8] L. Li, Y. Yu, S. Bai, Y. Hou and X. Chen, "An Effective Two-Step Intrusion Detection Approach Based on Binary Classification and K-NN," *IEEE Access*, vol. 6, pp. 12060-12073, Dec. 2017. Article (CrossRef Link)

[9] F. Jiang, Y. Sui and C. Cao, "An Incremental Decision Tree Algorithm Based on Rough Sets and Its Application in Intrusion Detection," *Artificial Intelligence Review*, vol. 40, no. 4, pp. 517-530, Dec. 2013. Article (CrossRef Link)

[10] Q. Yang, H. Fu and T. Zhu, "An Optimization Method for Parameters of SVM in Network Intrusion Detection System," in *Proc. of International Conference on Distributed Computing in Sensor Systems*, pp. 136-142, May. 26-28, 2016. Article (CrossRef Link)

[11] F. Gumus, C. Okan Sakar, Z. Erdem and O. Kursun, "Online Naive Bayes Classification for Network Intrusion Ddetection," in *Proc. of IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 670- 674, Aug. 17-20, 2014. Article (CrossRef Link)

[12] L. Koc and A. D. Carswell, "Network Intrusion Detection Using a HNB Binary Classifier," in *Proc. of 17th UKSim-AMSS International Conference on Modelling and Simulation*, pp. 81-85, Mar. 25-27, 2015. Article (CrossRef Link)

[13] X. Yin, Y. Zhang and X. Chen, "A Binary-Classification Method Based on Dictionary Learning and ADMM for Network Intrusion Detection," in *Proc. of International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pp. 326-333, Oct. 12-14, 2017. Article (CrossRef Link)

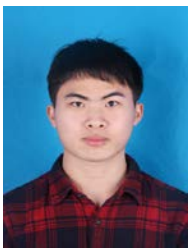
[14] Y. Y. Chung and N. Wahid, "A Hybrid Network Intrusion Detection System using Simplified Swarm Optimization (SSO)," *Applied Soft Computing*, vol. 12, no. 9, pp. 3014-3022, Sept. 2012. Article (CrossRef Link)

[15] K. C. Khor, C. Y. Ting and S. Phon-Amnuaisuk, "A Cascaded Classifier Approach for Improving Detection Rates on Rare Attack Categories in Network Intrusion Detection," *Applied Intelligence*, vol. 36, no. 2, pp. 320-329, Mar. 2012. Article (CrossRef Link)
[16] K. Atefi, S. Yahya, A. Y. Dak and A. Atefi, "A Hybrid Intrusion Detection System Based on Different Machine Learning Algorithms," in *Proc. of 4th International Conference on Computing and Informatics*, pp. 312-320, Aug. 28-30, 2013.
[17] G. R. Kumar, N. Mangathayaru and G. Narasimha, "An Improved K-means Clustering Algorithm for Intrusion Detection Using Gaussian Function," in *Proc. of International Conference on Engineering & Mis*, pp. 1-7, Sept. 24-26, 2015. Article (CrossRef Link)
[18] N. Liu and J. Zhao, "Intrusion Detection Research Based on Improved PSO and SVM," in *Proc. of International Conference on Automatic Control and Artificial Intelligence*, pp. 1263-1266, Mar. 3-5, 2012. Article (CrossRef Link)
[19] M. Y. Su, "Using Clustering to Improve the KNN-based Classifiers for Online Anomaly Network Traffic Identification," *Journal of Network Computer Applications*, vol. 34, no. 2, pp. 722-730, Mar. 2011. Article (CrossRef Link)
[20] C. Guo, Y. Zhou, Y. Ping, S. Luo, Y. Lai and Z. Zhang, "Efficient Intrusion Detection Using Representative Instances," *Computers & Security*, vol. 39, Part B, pp. 255-267, Nov. 2013. Article (CrossRef Link)
[21] F. Amiri, M. R. Yousefi, C. Lucas, A. Shakery and N. Yazdani, "Mutual Information-based Feature Selection for Intrusion Detection Systems," *Journal of Network Computer Applications*, vol. 34, no. 4, pp. 1184-1199, Jul. 2011. Article (CrossRef Link)
[22] Y. Li, J. Wang, Z. Tian, T. Lu and C. Young, "Building Lightweight Intrusion Detection System Using Wrapper-based Feature Selection Mechanisms," *Computers & Security*, vol. 28, no. 6, pp. 466-475, Sept. 2009. Article (CrossRef Link)
[23] F. Kuang, S. Zhang, Z. Jin and W. Xu, "A Novel SVM by Combining Kernel Principal Component Analysis and Improved Chaotic Particle Swarm Optimization for Intrusion Detection," *Soft Computing*, vol. 19, no. 5, pp. 1187- 1199, May. 2015. Article (CrossRef Link)
[24] C. F. Tsai and C. Y. Lin, "A Triangle Area Based Nearest Neighbors Approach to Intrusion Detection," *Pattern Recognition*, vol. 43, no. 1, pp. 222-229, Jan. 2010. Article (CrossRef Link)
[25] W. Lin, S. Ke and C. Tsai, "CANN: An Intrusion Detection System Based on Combining Cluster Centers and Nearest Neighbors," *Knowledge-Based Systems*, vol. 78, pp. 13-21, Apr. 2015. Article (CrossRef Link)
[26] X. Wang, C. Zhang and K. Zheng, "Intrusion Detection Algorithm Based on Density, Cluster Centers, and Nearest Neighbors," *China Communications*, vol. 13, no. 7, pp. 24-31, Jul. 2016. Article (CrossRef Link)
[27] K. Peng, V. C. M. Leung and Q. Huang, "Clustering Approach Based on Mini Batch Kmeans for Intrusion Detection System Over Big Data," *IEEE Access*, vol. 6, pp. 11897-11906, Feb. 2018. Article (CrossRef Link)
[28] D. Sculley, "Web-Scale K-means Clustering," in *Proc. of 19th International Conference on World Wide Web*, pp. 1177-1178, Apr. 26-30, 2010. Article (CrossRef Link)
[29] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, "A Detailed Analysis of the KDD CUP99 Data Set," in *Proc. of IEEE International Conference on Computational Intelligence for Security and Defense Applications*, pp. 1-6, Jul. 8-10, 2009. Article (CrossRef Link)
[30] A. Howard, "Elementary Linear Algebra," *7nd Edition, John Wiley & Sons*, pp. 170-171, 1994.
[31] L. Bottou and Y. Bengio, "Convergence Properties of The K-means Algorithms," in *Proc. of 7th International Conference on Neural Information Processing Systems*, pp. 585-592, 1995. Article(CrossRef Link)
[32] S. Manocha and M. A. Girolami, "An Empirical Analysis of the Probabilistic K-Nearest Neighbour Classifier," *Pattern Recognition Letters*, vol. 28, no. 13, pp. 1818-1824, Oct. 2007. Article (CrossRef Link)
[33] D. Arthur, "K-means++: The Advantages of Careful Seeding," in *Proc. of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027-1035, Jan. 07-09, 2007. Article (CrossRef Link)

[34] P. A. Flach, "The Geometry of ROC Space: Understanding Machine Learning Metrics through ROC Isometrics," in *Proc. of the Twentieth International Conference on Machine Learning*, pp. 194-201, Aug. 21-24, 2003.
[35] T. Fawcett, "An Introduction to ROC Analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861-874, Jun. 2006. Article (CrossRef Link)

Qiuhua Wang received her B.S. and M.S. degrees in communication engineering from Liaoning Technical University, Fuxin, China, in 2000 and 2003, respectively. She received her Ph.D. degree in communications and information systems from Zhejiang University, Hangzhou, China, in 2013. Now, she is an Associate Professor of the School of Cyber Security, Hangzhou Dianzi University. Her current research interests include network security, security issues in wireless networks, key management and physical layer security, etc.

Xiaoqin Ouyang received her B.S. degree from Hangzhou Dianzi University (HDU), Hangzhou, China, in 2016. She is currently pursuing the master degree in Communication Engineering in Hangzhou Dianzi University. Her research areas include information security, network security and intrusion detection based on data mining, etc.



Jiacheng Zhan received his B.S. degree from Hangzhou Dianzi University (HDU), Hangzhou, China, in 2016. He is currently pursuing the master degree in Communication Engineering in Hangzhou Dianzi University. His research interests include network security, privacy protection and security issues in wireless sensor network, key agreement for wireless sensor network, Zigbee protocol, and security in wireless sensor network, etc.